UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/686,628	10/12/2000	Paul J. Hinker	06502.0302-00	6118

60667 7590 07/18/2006

SUN MICROSYSTEMS/FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
VO, TED T	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 07/18/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/686,628	HINKER, PAUL J.	
	Examiner	Art Unit	
	Ted T. Vo	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 May 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☒ Claim(s) *20* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *5/1/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed as a RCE on 05/01/2006.

Claims 17-20 are newly added.

Claims 1-20 are pending in the application.

Response to Arguments

2. Applicant's arguments in the remarks filed on 15/01/2006 have been fully considered.

With regards to the arguments under 112 second paragraph, the rejection is withdrawn.

With regard to the arguments to the date publication of the prior arts: The sources of these references are named on the references and citation given under US form 1449, as "HP Invent", or go online: http://h21007.www2.hp.com/dspp/files/unprotected/64bitAppDev.pdf, and "Microsoft" online library. It should be known that both "HP Invent", and "Microsoft" are public accessible. Applicants could refer to these companies/corporations for more information.

See MPEP 715: Any printed publication or **activity dated prior to an applicant's or patent owner's effective filing date**, or any domestic patent of prior filing date, which is in its disclosure pertinent to the claimed invention, is available for use by the examiner as a reference, either basic or auxiliary, in the rejection of the claims of the application or patent under reexamination. In addition, patent application publications and certain international application publications having an effective prior art date prior to the application being examined may be used in a rejection of the claims. See MPEP § 706.02(a) and § 2136 - § 2136.03.

Furthermore, 102 statute says the following: "A person shall be entitled to a patent **unless** --
(a) the invention was **known** or **used** by others in this country, .., before the invention thereof by the applicant for a patent.
(b) the invention was patented or ...in **public use**... in this country, ..."
In light of the statute, it is irrelevant whether or not the HP/Microsoft publication is not a **printed** (even though printable) publication because the statute says that if the invention was **known** - 102(a) - or **in public use** - 102(b) - then the invention is not patentable in view of these references.

With regards to the arguments to the rejections under 102, where applicants alleged Coutant does not participates their claims.

Examiner disagrees. Based on the Applicants' arguments, they alleged that Microsoft or HP Packard does not teach 32-bit to 64-bit conversions. Based on the arguments, Applicant alleged that adding a subroutine in a file is a novel feature.

As noted in the prior office action, conversion of 2^n-bit to 2^{n+1}-bit is only conforming to the bit increase of an up-to-date technology. For example, a parameter in a particular programming run under 32-bit architecture some how unfits to an up-to-date 64-bit architecture. It requires a conversion. The conversion has been done by HP and Microsoft as given in the references. Manual acts such as adding deleting creating an interface file, a compared table are common. Moreover, with an ordinary skill, given a computer with standard windows, and a text editor, he can **create** an interface, he can **store** information in the interface file, he can **add** parameters in the file. It should be noted that these acts only manual acts. The user can insert in the file the directionality or parameters such as in the compared tables given in the Coutant's reference. It should be noted that the characters like int, float, long, etc, are parameter types, the dimension is such as 32 or 64, etc., and the compared table is a mere interface information/file provided to the user to identify the differences for conversion.

Coutant is clearly showing the conversion in this manner and using the stubs in order to port all the source program specification under 32-bit into the 64-bit. On the other hand, nowhere has been addressed or point out in the Applicants' remarks with the claimed novel features, but instead contending the elements that be performed with manual activities as the differences. The fact is that the scope of the claims are broadened that also covers **manual activities**, where Coutant reference suggests both dynamic and manual acts of the conversion in the compared table and the stubs. Therefore, in term patentability, claimed limitations that are covered by manual acts would not be patentable.

With regards to argument to Claim 3, where claim is only a preemption of conversion, nowhere a novel feature is recited. The fact is that 64-bit porting is known as the conversion of 32-bit source code into 64-source code. This is very common. The phases such as "receiving 32-bit source code", "based on the statements in the 32-bit interface file" etc., are only preemption from the manual acts. The compared tables in the reference clearly represent the interface files used by a user for 64-bit porting. The Coutant diagrams, tables, figures appear to cover the claim.

With regards to the arguments to disclosure of Microsoft where Applicants alleged the Microsoft is not a qualified prior art and does not teach 64-bit conversion.

Examiner disagrees. For an argument related to the qualified prior art, see above cited MPEP 715. The clear fact is that Microsoft has done since 1998 or before. The rejection is based on what the content provided in that reference and common knowledge. The term "last update" indicates as a version seen very common in printing publications. Moreover, the conversion from 32 bit to 64 bit is only conforming to a standard, and conversion of 2^n-bit to 2^{n+1}-bit will happen when extends bit performance of an architecture.

With regards to the arguments that Microsoft does not disclose their claims, Applicants should be directed to the whole reference, particularly "stubs" used in the reference.

Applicants' claims are broad, such broad claims is only a preemption of the words "conversion", "porting". Where the detailed discussion in the Microsoft clearly cover all generic conversion concept.

Many other Arguments have been considered but not persuasive because the Applicants' argument fail to point out the novel features in the Claims in accordance to MPEP 714.04.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

4. Claims 1-19 are rejected under 35 U.S.C. 102(a) as being anticipated by Coutant, "64-Bit Application Development for PA-RISC & IA-64", 3-2000.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Coutant discloses,

A method in a data processing system containing source code with a subprogram having at least one of an integer non-scalar parameter and a logical non-scalar parameter, the method comprising:

creating an interface file for the subprogram in the source code; storing in the interface file a definition of the subprogram; (See 64-bit programming model in p. 3-4)

adding to the interface file a directionality of at least one of the integer parameter (e.g. the type/declaration required by programming syntax such as "int") *and the logical parameter* (e.g. the type/declaration required by programming syntax such as "long") *based on comments* (a table shown in p. 3 has means of comment) *in the source code; adding to the interface file a parameter size along each dimension* (referred to 32-bit or 64-bit) *of at least one of the integer parameter and the logical parameter* (See 64-bit programming model in p. 3-4; See p. 11, 'linkage table', and see p.14, it shows a conversion that checks the types when mapped from 33-bit code to 64-bit code);

and reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters (See p.5-6).

As per Claim 2: Coutant discloses, *The method of claim 1, wherein the source code is 32-bit code and wherein the method further includes the step of invoking the 64-bit code from 32-bit code* (See p.5-6).

As per Claim 17: Coutant discloses claim 17: Based on the view of the compared table, and corresponding parameters, a user, with basis acts provided by computer environments, can "*determining whether the at least one parameter has input directionality, output directionality, or input or directionality; adding to the interface file statements based on the determined directionality*".

It should be noted that with manual acts a user can read the compared table that defines the directionality of the conversion and he can use the computer's file systems to add information in a file.

As per Claims 18: Coutant discloses, "*adding to the interface file statements indicating a number of dimensions of at least one parameter and a number of elements in each dimension.* E.g. see table in p. 3,

As per Claim 3: Coutant discloses, *A method in a data processing system, comprising the steps of:*

receiving 32-bit source code;

generating, from the 32-bit source code, a 32-bit interface file including statements describing characters of parameters in the 32-bit source code; and

automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64 bit code (This claimed limitation recites broadly a generation of 32-bit to 64 bit conversion stub See p.5-7: The 32-bit runtime used parameter relocation stubs to match caller and callee (p. 7)).

As per Claim 4: Coutant discloses,

The method of claim 3, wherein the 32-bit source code includes at least one of an integer parameter and a logical parameter and wherein the automatically generating step further includes the steps of:

determining whether the at least one of an integer and logical parameter has input directionality, output directionality, or input and output directionality; and

inserting into the 32-bit interface file code generator statements corresponding to the determined directionality of the at least one parameter.

(See p. 3-4 and p.5-6).

As per Claim 19: Coutant discloses, "*wherein generating a 32-bit interface file includes invoking an interface generator that: scans the 32-bit source code and creates the interface and create the interface file according to a definition;* See the compared tables in p. 3 or 4.

and adds to the interface file the statements describing characteristics of the parameters by parsing the 32-bit source code". For example, see statements, In LP64, Long and pointer are 64 bits long; All other base type are the same as ILP32 – these suggest a user how to add to the table (*the interface file*) used for comparing in a conversion.

As per Claim 5: Coutant discloses,

A data processing system, comprising:

a storage device, comprising:

source code with a subprogram having at least one of an integer and logical parameter;

an interface generator that reads the subprogram and that generates an interface file with indications of characteristics of the parameter; and

a stub generator that reads the interface file and that generates a stub for the subprogram by using the characteristics (p. 3-4, where the 'type' is identified as characteristics for the developing of 32-bit code and 64-bit code in conversing/porting), *wherein the stub receives a set of parameter values* (p. 3-4: i.e., expression in ILP32), *generates the values for the required parameters from the received set of parameter values, and invokes the subprogram with the values for the parameters; and*

a processor for running the interface generator and the stub generator (Whole claim is referred to p. 3-4 and p.5-6, as addressed in Claim 1 because Claim 5 is the system, typically a computer, that performs the step of Claim 1).

As per Claim 6: Coutant discloses,

The data processing system of claim 5, wherein the source code contains comments indicating the characteristics of the parameter (comments are part of instructions/code used in major programming languages).

As per Claim 7: Coutant discloses,

The data processing system of claim 6, wherein the characteristics include an indication of a conditional value for at least one of the required parameters (It should be noted that *conditional value for at least one of the required parameters* is mere instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions).

As per Claim 8: Coutant discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters is used to contain a return value.* (It should be noted that *an indication of whether at least one of the required parameters is used to contain a return value* is mere syntax of instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions).

As per Claim 9: Coutant discloses, *The data processing system of claim 6, wherein the characteristics include a directionality of at least one of the required parameters* (It should be noted that *a directionality of at least one of the required parameters* is mere syntax of instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions)

As per Claim 10: Coutant discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters required a multidimensional variable* (It should be noted that *an indication of whether at least one of the required parameters required a multidimensional variable* is mere syntax of instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions; i.e. array etc.)

As per Claim 11: Coutant discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether a size of at least one of the required parameters is based on another one of the required parameters* (It should be noted that *an indication of whether a size of at least one of the required parameters is based on another one of the required parameters* is mere syntax of instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions).

As per Claim 12: Coutant discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters is a work space parameter* (It should be noted that *an indication of whether at least one of the required parameters is a work space parameter* is mere syntax of instruction/code used in major programming languages. Further see p. 3-4, refer to 'type' and see the 'stub' and using programming instructions).

As per Claim 13: Coutant discloses, *A computer-readable medium containing instructions for controlling a data processing system to perform a method comprising the steps of:*

receiving 32-bit source code;

generating, from the 32-bit source code, a 32-bit interface file including statements describing characters of parameters in the 32-bit source code; and

automatically generating a 32-bit interface to 64-bit source code based on the statements in the interface file (First of all, any conversion that reads an interface will base on the statement in the interface file.

Therefore, based on this claimed limitation, it merely recites a generation of 32-bit to 64 bit conversion stub without further limitation to make the conversion be distinct from a generic statement "conversion". Coutant discloses conversion. See p.5-7: The 32-bit runtime used parameter relocation stubs to match caller and callee (p. 7)).

As per Claim 14: Coutant discloses,

The computer-readable medium of claim 13, wherein the 32-bit source code has a subprogram with a parameter and wherein generating a 32-bit interface file includes:

determining whether the parameter in the subprogram has input directionality, output directionality, or input and output directionality; and

inserting into the 32-bit interface file code generator statements corresponding to the determined directionality of the parameter in the subprogram.

(See p. 3-4, *interface for the subprogram*).

As per Claim 15: Coutant discloses, *A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code with a subprogram having a parameter, the method comprising the steps of:*

reading the source code;

generating from the source code in interface file including characteristics of the parameter; and

generating, based on the characteristics of the parameter, a stub routine that invokes the subprogram and that facilitates use of at least one of a converted integer and logical parameter. (The recitation is a mere conversion based on the characteristics of the parameter. See the compared table and see p. 5-6).

As per Claim 16: Coutant discloses, *A data processing system comprising:*

means for receiving 32-bit source code;

means for generating, from the 32-bit source code, a 32-bit interface including statements describing characteristics of parameters in the 32-bit source code; and

means for automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit stub to the 32-bit source code. (Recitation is a mere conversion. See p.3-4, and p. 5-6).

5. Claims 1-16, 17-18, 19, are rejected under 35 U.S.C. 102(b) as being anticipated by Microsoft, "Microsoft Interface Definition Language (MIDL): 64-Bit Porting Guide" 8-1999.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Microsoft discloses,

A method in a data processing system containing source code with a subprogram having at least one of an integer non-scalar parameter and a logical non-scalar parameter, the method comprising:

creating an interface file for the subprogram in the source code; storing in the interface file a definition of the subprogram; (See p.1-2, three bold dots. See p. 17, IDL files, conversing 32-bit to 64-bit).

adding to the interface file a directionality of at least one of the integer parameter and the logical parameter based on comments in the source code;

adding to the interface file a parameter size along each dimension of at least one of the integer parameter and the logical parameter; (See whole reference, particularly referring on the syntax of subroutine calls and Handle Types of MIDL. See text under "Issues", p.1-2. See a typical IDL setup used in the interface definition shows adding parameter lists, p.12);

and reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters (See p.15, when a compiler reading IDL interface definition file written under MIDL, it calls a 64-bit stub generation, for example, "As of Summer 1999, (i.e. Windows 2000 Pro RCx releases), the 64b type libraries are supported by mapping 32b*.TLB files...").

As per Claim 2: Microsoft discloses, *The method of claim 1, wherein the source code is 32-bit code and wherein the method further includes the step of invoking the 64-bit code from 32-bit code* (See whole reference, referring to terms "32-bit", "64-bit").

As per Claim 17: Microsoft discloses claim 17: for example, "*determining whether the at least one parameter has input directionality, output directionality, or input or directionality*; see statements in page 2: . New data types are introduce. . Old data types are changes in some way – These statements are

determining whether the at least one parameter has input directionality, output directionality, or input or directionality.

For example, *adding to the interface file statements based on the determined directionality*, see the statement mentioned above; i.e. Microsoft will base on the data type differences of 32-bit and 64-bit in order to add to the MIDL files. It should be noted that the above claim can be manually performed by a user.

As per Claims 18: Microsoft discloses, "adding to the interface file statements indicating a number of dimensions of at least one parameter and a number of elements in each dimension. For example, Microsoft says "int3264 has been added to the 64-bit compiler" (p. 15) – It means that "adding" thing in a file is only a common activity of a user.

As per Claim 3: Microsoft discloses, *A method in a data processing system, comprising the steps of:*

receiving 32-bit source code;

generating, from the 32-bit source code, a 32-bit interface file including statements describing characters of parameters in the 32-bit source code

(See passage in p. 17, You can use the first solution, or, you can set up the build separately for 64-bit and 32-bit, using separate 32/64-bit IDL files. In this way, you can have DWORD on 32-bit platform and ULONG64 on 64-bit platform. Or, you can use DWORD_PTR, once the optimization for ULONG_PTR/DWORD_PTR to remote 8 bytes when talking between 64-bit processes, and 4 bytes otherwise is supported. Examiner note: IDL file is Interface Definition Language file); and

automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64 bit code,

(see passage in p. 11-12, (started at the last line in p. 11), For a standard RPC interface, assume that the routine **f1** in an interface **IFace** requires conversions between the user arguments and what is actually transmitted. The following example is a typical IDL setup:
[local]
long f1 (<users parameter list>);

[call_as(f1)]
long Remf1 (<remotable parameter list>);

The specification indicates that instead of f1, Remf1 should be used when remoting. However, the specification would still cause the generated header file to define the interface using the definition of f1 (making the consistent old API interface for C linkage). Yet it would also provide stubs for marshaling

Remf1. The prototype for Remf1 is also generated to the h file. What are not generated anymore are the stubs for f1.
The user needs to write client and server stubs for f1 that will serve as the wrappers performing the desired function: verification, conversion, and so forth. These are the f1 routine on the client side and the Rem1 routine on the server. The client app would call f1, and f1 would call Rem1 (that is the MIDL generated stub) to marshal arguments. On the server, RPC would unmarshall Remf1 arguments, and then invoke the Rem1 wrapper provided by the user, which in turn would call the original f1).

As per Claim 4: Microsoft discloses,

The method of claim 3, wherein the 32-bit source code includes at least one of an integer parameter and a logical parameter and wherein the automatically generating step further includes the steps of:

determining whether the at least one of an integer and logical parameter has input directionality, output directionality, or input and output directionality; and

inserting into the 32-bit interface file code generator statements corresponding to the determined directionality of the at least one parameter. See the 32/64-bit IDL files, cited within "Frequently Asked Questions", p. 17.

As per Claim 19: Microsoft discloses, "*wherein generating a 32-bit interface file includes invoking an interface generator that: scans the 32-bit source code and creates the interface and create the interface file according to a definition; and adds to the interface file the statements describing characteristics of the parameters by parsing the 32-bit source code*". See Compiler Modes for 64-bit Platforms and Integral Types: __int32, __int64, __int3264, in p. 15. See 32/64-bit IDL files, cited within "Frequently Asked Questions", p. 17.

As per Claim 5: Microsoft discloses,

A data processing system, comprising:

a storage device,

(a computer used by Microsoft)

comprising: source code with a subprogram having at least one of an integer and logical parameter; (A program that is used for 32-bit/64-bit conversion)

an interface generator that reads the subprogram and that generates an interface file with indications of characteristics of the parameter; (See the Compiler cited with p. 15, and see IDL files cited within "Frequently Asked Questions, p. 17)

and

a stub generator that reads the interface file and that generates a stub for the subprogram by using the characteristics wherein the stub receives a set of parameter values, generates the values for the required parameters from the received set of parameter values, and invokes the subprogram with the values for the parameters; (See 64-bit Stub Generation Model, cited within p. 15)

and a processor for running the interface generator and the stub generator (referring to the computer that runs the conversion).

As per Claim 6: Microsoft discloses,

The data processing system of claim 5, wherein the source code contains comments indicating the characteristics of the parameter (i.e. IDL files, or see Integral Types: __int32, __int64, __int3264, in p. 15).

As per Claim 7: Microsoft discloses,

The data processing system of claim 6, wherein the characteristics include an indication of a conditional value for at least one of the required parameters (It should be noted that *conditional value for at least one of the required parameters* is mere instruction/code used in major programming languages. See Integral Types: __int32, __int64, __int3264, in p. 15. See two dotted statements p. 2).

As per Claim 8: Microsoft discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters is used to contain a return value.* (It should be noted that *an indication of whether at least one of the required parameters is used to contain a return value* is mere syntax of instruction/code used in major programming languages. For example see the program in p. 17).

As per Claim 9: Microsoft discloses, *The data processing system of claim 6, wherein the characteristics include a directionality of at least one of the required parameters* (It should be noted that *a directionality*

of at least one of the required parameters is mere syntax of instruction/code used in major programming languages. See Integral Types: __int32, __int64, __int3264, in p. 15).

As per Claim 10: Microsoft discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters required a multidimensional variable* (It should be noted that *an indication of whether at least one of the required parameters required a multidimensional variable* is mere syntax of instruction/code used in major programming languages. See Integral Types: __int32, __int64, __int3264, in p. 15).

As per Claim 11: Microsoft discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether a size of at least one of the required parameters is based on another one of the required parameters* (It should be noted that *an indication of whether a size of at least one of the required parameters is based on another one of the required parameters* is mere syntax of instruction/code used in major programming languages. Referring to 'type' and 'stub' 'IDL files', used in the Microsoft reference).

As per Claim 12: Microsoft discloses, *The data processing system of claim 6, wherein the characteristics include an indication of whether at least one of the required parameters is a work space parameter* (It should be noted that *an indication of whether at least one of the required parameters is a work space parameter* is mere syntax of instruction/code used in major programming languages. Referring to 'type' and 'stub' 'IDL files', used in the Microsoft reference).

As per Claim 13: Microsoft discloses, Microsoft discloses an interface definition that performs porting 32-bit into 64-bit that covers the limitation:

A computer-readable medium containing instructions for controlling a data processing system to perform a method comprising the steps of:

receiving 32-bit source code;

generating, from the 32-bit source code, a 32-bit interface file including statements describing characters of parameters in the 32-bit source code; and

automatically generating a 32-bit interface to 64-bit source code based on the statements in the interface file.

Microsoft discloses the conversion of 32-bit source code, used in generating 32-bit Windows application, to 64-bit Windows (See Summary in p. 1: This summary clearly address more than the claimed limitation above).

As per Claims 14-16:

The functionality of each of independent Claims 14-16 corresponds to functionality of Claim 13. Claims 14-16 have the same rejection as set forth in Claims 13 in regard to the teaching of Microsoft.

Allowable Subject Matter

6. Claim 20 is objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Conclusion

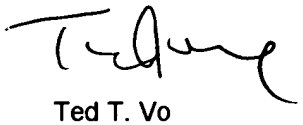
7. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (571) 272-3706. The examiner can normally be reached on 8:00AM to 4:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708.

The facsimile number for the organization where this application or proceeding is assigned is the Central Facsimile number **571-273-8300**.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Ted T. Vo
Primary Examiner
Art Unit 2191
July 07, 2006